DO NOT ENTER /Ben C Wang/ August 8, 2008

CERTIFICATE OF TRANSMISSION

I hereby certify that this correspondence is being transmitted by either submission using the EFS WEB submission system, fax to the U.S. Patent and Trademark office to fax number 571-273-8300, or is being deposited with the United States Postal Service as first class mail in an envelope addressed to: Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 to on July 29, 2008.

/Brian C. Kunzler/
Attorney for Applicants

PATENT
Docket No. TUC920030135US1

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Stephen La Roux Blinick et al.	
Serial No.:	10/717,941	
Confirm. No.:	9013	Group Art Unit: 2192
Filed:	November 20, 2003	
For:	APPARATUS, SYSTEM, AND METHOD FOR UPDATING AN EMBEDDED CODE IMAGE	
Examiner:	Ben C. Wang	

REQUEST FOR CONTINUED EXAMINATION

Mail Stop RCE
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

Dear Examiner:

In response to the Final Office Action mailed May 29, 2008, Applicants respectfully request the continued examination of the present application in view of the following amendments and remarks.

AMENDMENTS TO THE CLAIMS

Please replace the claims with the following listing of claims:

1. (Currently Amended) An apparatus for updating a code image, comprising:

a loader configured to load a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image;

a bootstrap module, within the new code image, configured to reconcile the incompatibilities by changing an initialization order; and

a copy module configured to copy the new code image into the memory space occupied by the old code image.

2. (Original) The apparatus of claim 1, wherein the old code image is updated substantially concurrent with normal execution of transactions by the apparatus.

3. (Original) The apparatus of claim 1, further comprising an initialization module configured to initiate execution of a run-time segment of the new code image.

4. (Previously Presented) The apparatus of claim 1, wherein the copy module copies the new code image into the memory space after the bootstrap module reconciles the incompatibilities.

5. (Currently Amended) The apparatus of claim 1, wherein the logic module accesses the version information for the old code image and version information for the new code image to identify an incompatibility based at least in part on a difference between the version information.

6. (Previously Presented) The apparatus of claim 5, wherein the bootstrap module is configured to reconcile the incompatibility by updating modules that interface with the new code image.

7. (Previously Presented) The apparatus of claim 1, wherein the bootstrap module is configured to reconcile incompatibilities by associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of the run-time segment of the new code image.

8. (Canceled)

9. (Original) The apparatus of claim 1, wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image.

10. (Currently Amended) An apparatus for updating a code image, comprising:

an update module configured to load a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

a logic module configured to identify incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image; and

a bootstrap module within the new code image that executes subsequent to the update module, the bootstrap module configured to reconcile the incompatibilities by changing an initialization order prior to copying the new code image into the memory space occupied by the old code image.

11. (Currently Amended) The apparatus of claim 10, wherein the bootstrap module is configured to reconcile incompatibilities based on the version information for the old code image and the new code image, and wherein a copy module is configured to copy the new code image over the old code image in after the incompatibilities have been reconciled.

12. (Original) The apparatus of claim 10, wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image.

13. (Currently Amended) A system that overlays an old code image with a new code image with minimal interruption of operations being performed by execution of the old code image, the system comprising:

a memory comprising an old code image and a buffer configured to store a new code image;

a processor executing instructions of the old code image to perform one or more operations, the processor configured to execute instructions of the old code image and the new code image;

a data structure configured to store an old code image pointer and a new code image pointer;

wherein, in response to an interrupt, the processor begins identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image and executing bootstrap code within the new code image, the bootstrap code configured to reconcile incompatibilities between the old code image and the new code image by changing an initialization order.

14. (Original) The system of claim 13, wherein the bootstrap code overlays the new code image in memory with the old code image in response to reconciliation of the incompatibilities.

15. (Original) The system of claim 14, wherein in response to the interrupt, the processor executes an update module of the old code image that loads the new code image into the buffer.

16. (Original) The system of claim 15, wherein the update module stores the old code image pointer and the new code image pointer in the data structure.

17. (Original) The system of claim 16, wherein the update module reads a new code image header identified by the new code image pointer to determine the location of the bootstrap code within the new code image.

18. (Currently Amended) The system of claim 17, wherein the bootstrap code <u>further</u> reconciles the incompatibilities by updating modules that interface with the new code image.

19. (Currently Amended) The system of claim 18, wherein the bootstrap code <u>further</u> reconciles the incompatibilities by associating persistent data of the old code image with the new code image.

20. (Currently Amended) A method for updating a code image, comprising:

loading a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image;

reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image;

copying the new code image into the memory space occupied by the old code image.

21. (Original) The method of claim 20, wherein the old code image is updated substantially concurrently with execution of regular computer operations.

22. (Original) The method of claim 20, further comprising initiating execution of a run-time segment of the new code image.

23. (Previously Presented) The method of claim 20, wherein the new code image is copied into the memory space after the incompatibilities are reconciled.

24. (Currently Amended) The method of claim 20, wherein identifying incompatibilities between the old code image and the new code image further comprises accessing capability information for the old code image and capability information for the new code image and identifying a difference between the capability information.

25. (Previously Presented) The method of claim 24, wherein reconciling the incompatibilities comprises updating modules that interface with the new code image based at least in part on ~~a~~ the difference between the capability information.

26. (Currently Amended) The method of claim 20, wherein reconciling the incompatibilities further comprises associating persistent data of the old code image with the new code image, such that the persistent data is available in response to execution of a run-time segment of the new code image.

27. (Canceled)

28. (Original) The method of claim 20, wherein at least one of the incompatibilities comprises a difference between data structures used by the old code image and data structures used by the new code image.

29. (Currently Amended) An apparatus for updating a code image, the apparatus comprising:

means for loading a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

means for identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image;

means for reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image; and

means for copying the new code image into the memory space occupied by the old code image.

30. (Currently Amended) An article of manufacture comprising a program storage medium readable by a processor and embodying one or more instructions executable by a processor to perform a method for updating a code image, the method comprising:

loading a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;

identifying incompatibilities between the old code image and the new code image from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image;

reconciling the incompatibilities by changing an initialization order using bootstrap code of the new code image; and

copying the new code image into the memory space occupied by the old code image.

REMARKS

Claims 1-4, 7, 10, 13, 20-22, 26, and 29-30 stand rejected under 35 U.S.C. 103(a) as unpatentable over US patent application publication 2004/0044997 by Talati (hereinafter Talati) in view of US patent 6,658,659 to Hiller (hereinafter Hiller). Claims 5, 6, 8, 9, 11, 12, 23-25, and 27-28 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati and Hiller in view of US patent 6,986,132 to Schwabe (hereinafter Schwabe).

AMENDMENTS TO THE CLAIMS

Applicants have amended claim 1 with the limitation "…identify incompatibilities between the old code image and the new code image<u> from version information, a difference in initialization requirements, and a difference in size and location between the old code image and the new code image</u> …" The amendment is well supported by the specification. See page 11, ¶ 38-39; page 19, ¶ 72.

Applicants have further amended claim 1 with the limitation "…reconcile the incompatibilities<u> by changing an initialization order</u> …" The amendment is well supported by the specification. See page 11, ¶ 41.

Claims 10, 13, 20, 29, and 30 are similarly amended. Claims 5, 11, 18, 19, 24 and 26 are amended to conform to amended claims. Claims 8 and 27 are canceled.

RESPONSE TO CLAIM REJECTIONS UNDER 35 U.S.C. § 103

Claims 1-4, 7, 10, 13, 20-22, 26, and 29-30 stand rejected under 35 U.S.C. 103(a) as unpatentable over Talati in view of Hiller. Claims 5, 6, 8, 9, 11, 12, 23-25, and 27-28 stand rejected under 35 U.S.C. § 103(a) as unpatentable over Talati and Hiller in view of Schwabe. Applicants respectfully traverse these rejections.

Independent claim 1 includes the limitations:

> a loader configured to load a new code image into a temporary memory location separate from a memory space occupied by and used by an old code image;
>
> a logic module configured to **identify incompatibilities** between the old code image and the new code image from version information, **a difference in initialization requirements, and a difference in size and location** between the old code image and the new code image;
>
> a bootstrap module, within the new code image, configured to **reconcile the incompatibilities by changing an initialization order**; and
>
> a copy module configured to copy the new code image into the memory space occupied by the old code image.

Independent claims 10, 13, 20, 29, and 30 include similar limitations. Applicants submit that neither Talati, Hiller, nor Schwabe disclose identifying incompatibilities from a difference in initialization requirements and a difference in size and location between an old code image and a new code image. Hiller teaches identifying incompatibilities using compatibility vectors. Hiller, col. 9, lines 22-28; fig. 3A-3C, fig. 5. Schwabe discloses identifying incompatibilities by verifying internal consistency, verifying consistency with an API definition file, and comparing first and second versions of the program. Schwabe, Abstract. However, Talati, Hiller, and Schwabe do not teach the elements of identifying incompatibilities from a difference in initialization requirements and a difference in size and location between an old code image and a new code image.

Applicants also submit that Talati, Hiller, and Schwabe do not teach reconciling incompatibilities by changing an initialization order. Hiller teaches reconciling incompatibilities by loading specified versions of software, but does not disclose reconciling incompatibilities by changing an initialization order.

Because Talati, Hiller, and Schwabe do not teach each element of independent claims 1, 10, 13, 20, 29, and 30, Applicants submit that claims 1, 10, 13, 20, 29, and 30 are allowable. Furthermore, dependent claims 2-7, 9, 11-12, 14-19, 21-26, and 28 are allowable at least due to their dependency from independent claims 1, 10, 13, and 20.

CONCLUSION

Applicants submit that the remarks and amendments put the present application in condition for allowance. In the event the Examiner finds any remaining impediment to the prompt allowance of any of these claims, which could be clarified in a telephone conference, the Examiner is respectfully urged to initiate the same with the undersigned.

Respectfully submitted,

/Brian C. Kunzler/

Brian C. Kunzler
Reg. No. 38,527
Attorney for Applicants

Date: July 29, 2008
Kunzler & McKenzie
8 East Broadway, Suite 600
Salt Lake City, UT 84111
Telephone (801) 994-4646
Fax (801) 531-1929